Improving Numerical Prediction With Qualitative Constraints

Dorian Šuc[1] and Ivan Bratko[1]

Faculty of Computer and Information Science, University of Ljubljana,
Tržaška 25, 1000 Ljubljana, Slovenia
{dorian.suc, ivan.bratko}@fri.uni-lj.si
Published in Proc. of European Conf. on Machine Learning, ECML 2003

Abstract. The usual numerical learning methods, that are primarily concerned with finding a good numerical fit to the data, often make predictions that do not correspond to the qualitative mechanisms in the domain of modelling or a domain expert's intuition. Consistency of numerical predictions with a given qualitative model is helpful when a numerical model is used for explanation of phenomena in the modelled domain, but can also considerably improve numerical accuracy. In this paper we present a novel approach to numerical machine learning called Qfilter. Qfilter is a numerical regression method that can take into account qualitative background knowledge to give *qualitatively faithful numerical prediction*. The results on a set of domains including population dynamics show considerable prediction accuracy improvements compared to the usual numerical learners. As qualitative domain knowledge is often available in practice, Qfilter's ability to exploit such knowledge should be beneficial in many applications.

1 Introduction

1.1 Qualitative Problems of Numerical Learning

Methods of numerical machine learning, such as regression tree learning and locally weighted regression (LWR), often make predictions that a knowledgeable user finds obviously incorrect. A domain expert finds such errors incorrect not so much in numerical, but in qualitative terms. Often there are a priori known qualitative constraints in the domain of application, and for numerical predictions to make sense, the predictions should be consistent with such constraints.

For example, consider a container filled with water. Let there be an open drain at the bottom of the container, and water is draining out. Suppose we want to make predictions about the amount of water at various times. Although exact numerical predictions of the amount may be hard, obviously these predictions will have to satisfy some qualitative constraints, such as: (1) the amount can never be negative, and (2) the amount of water in the container can never be increasing. Suppose that we have examples of measurements of the amount of

water in time, obtained from past behavior of the draining process that started at different initial amounts. We then use standard methods of numerical machine learning to make predictions of the amount at future times, starting with some new initial amount. Unfortunately, state-of-the-art numerical learning techniques will typically produce predictions that do not completely respect the above mentioned qualitative constraints even when the learning data is noise-free. Šuc et al. [1] give pertinent experimental results with the draining process, using M5 regression and model trees [2] and LWR [3] (implementation in Weka; [4]).

Such qualitative errors of numerical predictors are undesirable particularly because they make numerical results difficult to interpret. The underlying mechanism in the domain is usually best explained in qualitative terms. However, this is obscured by qualitative errors in numerical predictions.

1.2 Qfilter

In this paper we introduce a numerical learning method, called Qfilter. Qfilter accepts as input a set of numerical data points and a set of qualitative constraints, and performs numerical regression so that the predicted numerical values respect the given qualitative constraints. In a typical application of Qfilter, the qualitative constraints are provided by the domain expert as (qualitative) background knowledge. Such constraints are typically part of domain knowledge. For example, in biological modelling the growth rate of a population is qualitatively proportional to the current size of the population and to the amount of food available to the population. Another possibility of applying Qfilter is within the Q^2 learning, described below. In this context, qualitative constraints do not have to come from a domain expert.

In Section 2 we define the type of qualitative constraints and qualitative trees accepted by Qfilter. In Section 3 we describe the Qfilter algorithm. Section 4 presents experiments with Qfilter and a comparison with standard numerical prediction methods. Section 5 gives conclusions.

1.3 Relation of Qfilter to Q^2 Learning

To rectify the qualitative problems of numerical learning, Šuc et al. [1] proposed "qualitatively faithful quantitative learning", called Q^2 learning for short. In experiments with a complex industrial modelling problem, Q^2 learning not only improved the predictions qualitatively, but also numerically. Numerical predictions with Q^2 were considerably more accurate than those obtained with the mentioned numerical learning methods.

Q^2 learning consists of two stages:

1. Induce a qualitative model from numerical examples. Program QUIN ([5, 6]) that induces qualitative trees from numerical data can be used for this.
2. Transform a qualitative model induced in stage 1 into a quantitative model (i.e. a numerical function) that fits well the given numerical data and is consistent with the qualitative constraints in the qualitative tree. This transformation is called Q2Q transformation (qualitative-to-quantitative).

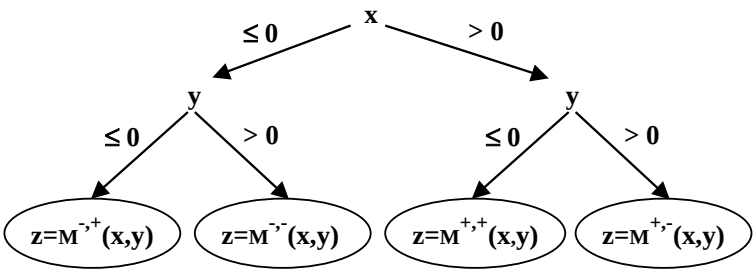

Fig. 1. A qualitative tree that describes the qualitative relations between class Z and attributes X and Y for the function $Z = X^2 - Y^2$. The rightmost leaf, applying when attributes X and Y are positive, says that Z is strictly increasing in its dependence on X and strictly decreasing in its dependence on Y.

The Q2Q transformation in [1] was based on piece-wise linear regression where these linear functions were determined heuristically using LWR on a grid of selected points. Although this method worked well in the experiments, it was ad hoc in that there was no guarantee that the so obtained heuristic functions would completely respect the qualitative constraints. The Qfilter approach introduced in this paper is a better founded and better performing method for Q2Q transformation.

2 Qualitative trees for knowledge representation

In this section we describe a formalism for the representation of qualitative background knowledge. We represent qualitative knowledge in the form of so-called qualitative trees that are described below and proved to be useful and understandable in several different applications [5, 7, 1]. In these applications qualitative trees were induced from numerical examples by program QUIN [5, 6]. In this paper we assume that they are given, e.g. defined by a domain expert, and study the advantages in terms of prediction accuracy.

Qualitative trees are similar to decision trees but model qualitative relations between the class and the attributes. As in decision trees, the internal nodes in a qualitative tree specify conditions that split the attribute space into subspaces. In a qualitative tree, however, each leaf specifies a region in the attribute space where some monotonicity constraints hold. These monotonicity constraints are represented by what we call *qualitatively constrained functions* (QCFs for short). A simple example of QCF is: $Y = M^+(X)$. This says that Y is a monotonically increasing function of X. In general, QCFs can have more than one argument. For example, $Z = M^{+,-}(X, Y)$ says that Z monotonically increases in X and monotonically decreases in Y. We say that Z is positively related to X and negatively related to Y. If both X and Y increase, then according to this constraint, Z may increase, decrease or stay unchanged. In such a case, a QCF cannot make an unambiguous prediction of the qualitative change in Z as explained below.

Figure 1 gives an example of a qualitative tree. This qualitative tree is a qualitative model of the function $Z = X^2 - Y^2$ and describes how Z qualitatively depends on attributes X and Y. The tree partitions the attribute space into four regions that correspond to the four leaves of the tree. A different QCF applies in each of the leaves. The QCF $Z = M^{+,-}(X, Y)$ applies in the rightmost leaf, where both X and Y are positive.

Qualitatively constrained functions are inspired by the qualitative proportionality predicates Q_+ and Q_- as defined by Forbus [4] and are also a generalization of the qualitative constraint M^+, as used in QSIM [9]. A QCF constrains the qualitative change of the class variable in response to the qualitative changes of the attributes. Namely, a QCF $M^{s_1,\ldots,s_m}$, $s_i \in \{+, -\}$ represents an arbitrary function $\Re^m \mapsto \Re$ with m continuous attributes that respects the qualitative constraints given by signs s_i. The qualitative constraint given by sign $s_i = +$ ($s_i = -$) requires that the function is strictly increasing (decreasing) in its dependence on the i-th attribute. We say that the function is *positively related* (negatively related) to the i-th attribute. $M^{s_1,\ldots,s_m}$ represents any function which is, for all $i = 1, \ldots, m$ positively (negatively) related to the i-th argument, if $s_i = +$ ($s_i = -$).

Note that the qualitative constraint given by sign $s_i = +$ only states that when the i-th attribute increases, the QCF will also increase, *barring other changes*. It can happen that a QCF with the constraint $s_i = +$ decreases even if the i-th attribute increases, because of a change in another attribute. For example, consider the behaviour of gas pressure in a container given by equation $Pres \times Vol/Temp = const$. We can express the qualitative behaviour of gas pressure by QCF $Pres = M^{+,-}(Temp, Vol)$. This constraint allows that the pressure decreases even if the temperature increases, because of a change in the volume. Notice however, that the qualitative behaviour of gas is not consistent with the constraint $Pres = M^+(Temp)$.

QCFs are concerned with qualitative changes. Qualitative change q_i in the i-th attribute is the sign of change in that variable. This can be either *positive*, *negative* or *zero* change. For simplicity, we ignore zero changes in the next paragraphs. QCF-prediction $P(s_i, q_i)$ is the qualitative change of the class variable predicted according to a single (i-th) attribute. QCF-prediction is positive if s_i and q_i are both positive or both negative, and is negative otherwise. *Qualitative ambiguity*, i.e. ambiguity in the class's qualitative change appears whenever there exist both positive and negative QCF-predictions according to different attributes. A qualitatively constrained function is *consistent* with a pair of examples if there exists an attribute whose QCF-prediction is equal to the class's qualitative change. We say that this example pair is QCF-consistent. A qualitatively constrained function is *consistent* with a set of examples if it is consistent with all possible example pairs, i.e. when all possible examples pairs are QCF-consistent.

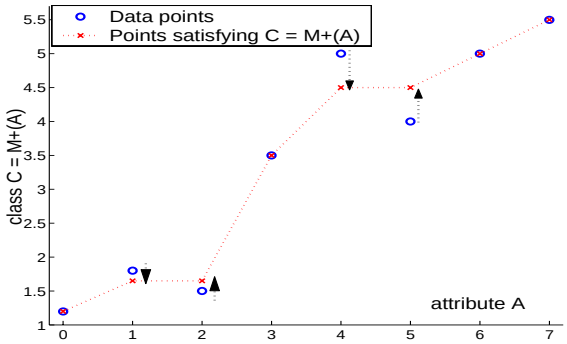

Fig. 2. Achieving consistency with QCF $C = M^+(A)$: since the QCF requires that class C is strictly increasing in attribute A, class values c_i (denoted by circles) are changed into $c_i + d_i$ (denoted by crosses) by minimizing the sum of squared changes d_i. The arrows denote the class changes d_i.

3 Qfilter

3.1 Idea and Example

Here we describe algorithm Qfilter that, given a set of examples and a qualitative tree, adjusts the class values in such a way that they are consistent with the qualitative tree. Namely, Qfilter is an optimization procedure that finds the minimal required quadratic changes in class values to achieve qualitative consistency with the qualitative tree. In one respect Qfilter can be viewed as a filter that smooths the data and removes the qualitative errors introduced by the measurement errors, but here we use it to remove the qualitative errors made by numerical predictors, as it will be explained later.

Let us first observe a simple example illustrated in Figure 2. We have eight examples (a_i, c_i), $i = 0, 1, .., 7$ described with the values of class C and attribute A. The examples are not consistent with the given QCF $C = M^+(A)$, because the QCF requires that $c_{i+1} > c_i$ which is violated at $i = 1$ and $i = 4$.

To achieve consistency with $C = M^+(A)$, class values should be changed into $c_i + d_i$, where the unknown parameter d_i denotes the change in i-th class value. Class changes d_i are constrained by QCF imposed inequalities: $c_{i+1} + d_{i+1} > c_i + d_i$ where $i = 0, 1, .., 6$. This gives the optimization problem that can be formulated in matrix notation by writing the inequalities as $\mathbf{A}\,\mathbf{d} > \mathbf{b}$, where $\mathbf{d}$ is a vector of unknown parameters d_i, vector $\mathbf{b}$ has elements $b_i = c_i - c_{i+i}$, and matrix $\mathbf{A}$ has elements $a_{i,i} = -1$, $a_{i,i+1} = 1$ and zeros elsewhere. Therefore finding minimal quadratic changes in class values that achieve consistency with a given QCF can be posed as the optimization problem:

$$\begin{aligned} &\text{find vector } \mathbf{d} \text{ that minimizes } \mathbf{d}^T\mathbf{H}\,\mathbf{d} \\ &\text{such that } \mathbf{A}\,\mathbf{d} > \mathbf{b} \end{aligned} \tag{1}$$

In the above formulation matrix $\mathbf{H}$ is the identity matrix. In general $\mathbf{H}$ can be changed to differently penalize the changes in class values. For example, we could

change $\mathbf{H}$ to require that the classes of examples that have higher confidence are changed less. The above stated optimization problem is a kind of *quadratic programme* and can be efficiently solved by a number of methods. We used a quadratic programming solver in Matlab [10–12]. Since the criterion function $\mathbf{d}^T\mathbf{H}\,\mathbf{d}$ with diagonal matrix $\mathbf{H}$ is a convex function, and because the linear constraints $\mathbf{A}\,\mathbf{d} > \mathbf{b}$ define a convex hull, any local minimum of the criterion function is a globally optimal solution.

Note that in the above formulation the values of attribute A are not mentioned at all. However the ordering of attributes values, i.e. $a_{i+1} > a_i$, was used together with QCF $C = M^+(A)$ to set the ordering of class values, i.e. $c_{i+1} > c_i$. A different ordering of attribute values would require a different ordering of class values, depending also on the QCF. When more than one attributes are used in a QCF, finding an appropriate ordering of class values is not as trivial as in the example above and is explained in the next section.

3.2 Details of the Qfilter Algorithm

Qfilter handles each leaf of a qualitative tree separately. It first splits the examples according to the qualitative tree and then change class values to achieve consistency with a QCFs in the corresponding leaf. As mentioned above, Qfilter uses the attributes' values and the QCF to find an appropriate ordering of class values, poses the optimization problem in the form given by Equation 1, and solves it by using quadratic programming methods. To explain how to find an appropriate ordering of class values we first define two useful terms, and then explain some properties of QCFs. Since in general not all of the attributes appear in a QCF, we call an attribute that appears in a QCF a *QCF-attribute*. We call a QCF that doesn't have a negative dependance on an attribute, i.e. is positively related with all QCF-attributes, a *pure-positive QCF*.

The first interesting QCF property is that an arbitrary QCF can be, by appropriate changes in attributes, replaced by a pure-positive QCF. It is easy to check that a QCF that is negatively related with attribute A_i and positively related with all other attributes, is equivalent to a pure-positive QCF, where the attribute A_i is replaced by $\hat{A}_i = -A_i$. For example QCF $M^{+,-}(A_1, A_2)$ is equivalent to QCF $M^{+,+}(A_1, \hat{A}_2)$, where $\hat{A}_2 = -A_2$. Therefore we can simply multiply by minus one (or any other negative number) all the negatively related QCF-attributes to get a pure-positive QCF. Actually multiplying by minus one all negatively related QCF-attributes is the first step in Qfilter. In the rest of this section we assume that a given QCF is a pure-positive QCF.

The second interesting QCF property is that a QCF defines a partial ordering of class values and that a pure-positive QCF is consistent with a set of examples if the class value of every example e is greater than the class values of every example from a set called $F_{sg}(e)$. More formally, we show that a pure-positive QCF is consistent with a set of examples if, and only if:

$$\forall e, f \in \text{Examples} : f \in F_{sg}(e) \Rightarrow c_e > c_f \tag{2}$$

where $F_{sg}(e)$ is the set of examples f_{sg} that are smaller than e in every QCF-attribute and there is no example f_s that is in every attribute smaller than e and in every QCF-attribute greater than f_{sg}. This is explained in the next paragraphs. We show this by using the QCF consistency criterion that requires that all possible example pairs are QCF-consistent. A first simplification is that we do not need to check all possible example pairs. A pure-positive QCF is consistent with a set of examples if every example e is QCF-consistent with all the examples f_s that are *smaller than e in every QCF-attribute*. The set of such examples is denoted by $F_s(e)$. Namely, example pairs of e and examples f_{amb} that are in one QCF-attribute grater than e, and in another QCF-attribute smaller than e are QCF-consistent with any pure-positive QCF. QCF-consistencies of example pairs of e and examples f_g, that are in all QCF-attributes grater than e, are checked when examples f_g are checked. Since the relation "is smaller than e in every QCF-attribute" is transitive, we need to check the consistency only for the "largest" examples from $F_{sg}(e) \subseteq F_s(e)$, i.e. examples $f_{sg} \in F_s(e)$ with the property that in F_s there is no example that is in every QCF-attribute greater than example f_{sg}. Since all the examples from $F_{sg}(e)$ are smaller than e in every QCF-attribute, a pure-positive QCF requires that the class value of example e is greater than the class value of every example from the set $F_{sg}(e)$, i.e. $\forall f \in F_{sg}(e) : c_e > c_f$.

The ordering of class values given by Equation 2 is used to set the inequality constraints, i.e. matrix $\mathbf{A}$ and vector $\mathbf{b}$ from Equation 1. For each $f \in F_{sg}(e)$ we add one (say i-th) constraint $c_e + d_e > c_f + d_f$, therefore vector $\mathbf{b}$ has elements $b_i = c_f - c_e$, and matrix $\mathbf{A}$ has elements $a_{i,f} = -1$, $a_{i,e} = 1$ and zeros elsewhere.

3.3 Qfilter for Numerical Prediction

The basic idea of using Qfilter for numerical prediction is to apply it, with a given qualitative tree, on predictions of an arbitrary numerical learner. A numerical predictor is usually trained on a set of learning examples, where "correct" class values are given. For this reason it is quite natural to provide the learning examples also to Qfilter. In this case Qfilter is supplied with the learning examples with "correct" class values together with test examples with predictions of class values. Qfilter then adjusts the class values of both learning and test examples to fit a qualitative tree. It is quite obvious that using also learning examples usually helps Qfilter. This is especially evident when adjusting a prediction of a test example that is close to some learning examples.

One possible improvement of Qfilter is to also use the confidence estimate in numerical prediction if it is provided by the numerical predictor. In this case, Qfilter would change the class values with higher confidence less, for the price of bigger changes of class values that have lower confidence. This is achieved by simply changing matrix $\mathbf{H}$ in Equation 1 from identity to a diagonal matrix with $h_{i,i} = w_i$, where weight w_i is computed from predictor's confidence estimate in i-th class value. Of course, the computation of weight w_i depends on the type and scale of confidence estimate, but would generally be smaller if a numerical predictor is more confident in i-th class prediction.

4 Experimental Results

Here we compare the numerical accuracy of locally weighted regression [3] (LWR) and Qfilter. Qfilter was used to adjust the LWR predictions according to given qualitative trees. We used a standard procedure to optimize LWR. Namely, LWR optimized the Gaussian kernel width that is used to weigh the neighbor examples according to the mean squared local cross validation error at the point of prediction.

We experimented with different learning set sizes and different noise in class variable. We used normally distributed zero-mean noise. Noise percentage p % means that the standard deviation of noise is $d_c\ p/100$ where d_c denotes the difference between maximal and minimal class value. First we describe experiments in four artificial domains and then experiments in a more complex population dynamics domain.

4.1 Artificial Domains

Here we describe experiments in four artificial domains. The first is the domain called *Quad* with attributes X and Y and class $Z = X^2 - Y^2$. Attributes X and Y are uniformly distributed between -10 and 10. Qfilter used LWR predictions and the qualitative tree given in Figure 1 and explained in Section 2.

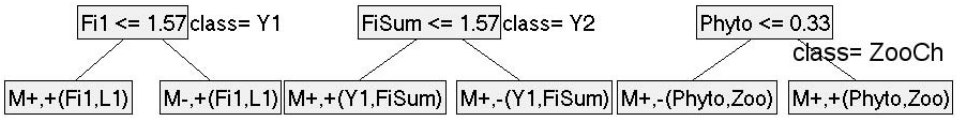

Fig. 3. Qualitative trees used with Qfilter in domains *RoboY1*, *RoboY2atrY1* and in the population dynamics domain *ZooChange*. Note that 1.57 in the first two trees is an approximation of $\frac{\pi}{2}$ where $\sin(\Phi)$ changes from an increasing to a decreasing function.

The second set of domains consists of three domains, called *RoboY1*, *RoboY2* and *RoboY2atrY1*. Here we model a planar two-link, two joint robot arm. The angle in the shoulder joint is denoted by Φ_1 and the angle in the elbow joint is denoted by Φ_2. Angle Φ_1 is between zero and π, while Φ_2 is between $-\pi/2$ and $\pi/2$. When the arm is in horizontal position Φ_1 and Φ_2 are both zero. The first link, i.e. the link from shoulder to elbow, is extendible with length L_1 ranging from 2 to 10. The second link has fixed length $L_2 = 5$. The first learning problem is to predict y-coordinate of the first link end, i.e. $Y1 = L_1 \sin(\Phi_1)$. This problem is called *RoboY1*. For Qfilter we used the qualitative tree given in Figure 3.

The second learning problem is to predict y-coordinate of the second link end, i.e. $Y2 = L_1 \sin(\Phi_1) + 5 \sin(\Phi_1 + \Phi_2)$. Here we helped the learners with a derived attribute $\Phi_{sum} = \Phi_1 + \Phi_2$, i.e. the deflection of the second link from the horizontal. We experimented with two versions of this learning problem. In domain

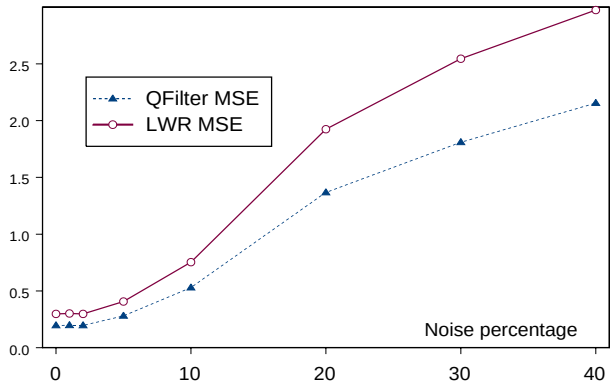

Fig. 4. Noise curve in domain *RoboY1* with 100 learning examples: on x- axis is noise percentage and on y-axis is LWR (line with circles) and Qfilter (dotted line with triangles) mean squared error.

RoboY2 we used the attributes L_1, Φ_1, Φ_2 and Φ_{Sum}. In domain *RoboY2atrY1* we also used the correct $Y1$ as an attribute. We generated examples where angles Φ_1 and Φ_2 and link length L_1 are uniformly distributed in their possible ranges. Qualitative tree used with domain *RoboY2atrY1* is given in Figure 3. Qualitative tree used with domain *RoboY2* has four leaves, with the same root node as qualitative tree for domain *RoboY2atrY1*, but with $Y1$ replaced by the qualitative tree for $Y1$.

We experimented with different learning set sizes and different noise in class variable. We used a test set of 200 examples without noise. Table 1 gives the comparison of LWR and Qfilter mean squared errors (MSE) with 100 learning examples and various noise levels. All the results are averages on 10 sets of randomly selected learning and test examples. With all four learning problems the improvement of Qfilter with respect to LWR is obvious. Qfilter usually reduces LWR MSE by more than 20 %. The MSE reduction usually increases with increased noise. Figure 4 shows a typical noise curve in domain *RoboY1*.

We also experimented with different learning set sizes. For an illustration, we give the results with learning from examples with no noise in domain *RoboY1* in Table 1. When we used only 10 or 20 learning examples the Qfilter reduction of error is relatively small, since none of the learners is able to generalize well from such a small learning set. But as the learning set increases, Qfilter can take advantage of given qualitative knowledge. After a certain learning set size, the reduction of error decreases with increasing learning set. However, the reduction in error is usually still visible even when we use relatively large learning set. Of course this depends on the difficulty of the domain. When a numerical learner gives predictions that are consistent with a given qualitative tree, Qfilter does not change them.

Table 1. Comparison of LWR and Qfilter accuracy. The first table gives MSE with 100 learning examples and various noise levels in all the described domains. Since the changes in zooplankton are small, the values of MSE given for the domain *ZooChange* are multiplied by 10^3. The second table gives MSE in domain *RoboY1* when different number of learning examples with no noise were used. All the results are averages on 10 sets of learning examples.

Domain name	class variable	no noise MSE LWR; Qfilter	5 % n. MSE LWR; Qfilter	20 % n. MSE LWR; Qfilter
Quad	$Z = X^2 - Y^2$	98.4 ; 84.7	149.3 ; 114.6	765.6 ; 554.7
RoboY1	$Y1 = L_1 \sin(\Phi_1)$	0.298 ; 0.196	0.407 ; 0.280	1.924 ; 1.367
RoboY2	$Y2 = L_1 \sin(\Phi_1) + 5 \sin(\Phi_{sum})$	2.618 ; 2.305	3.078 ; 2.612	6.823 ; 5.167
RoboY2atrY1	$Y2$ as above, using attr. $Y1$	0.940 ; 0.691	1.324 ; 0.968	3.665 ; 2.707
ZooChange	$ZooCh(t) = Zoo(t+1) - Zoo(t)$	0.015 ; 0.008	0.112 ; 0.102	2.269 ; 1.889

Domain name	20 learn. ex. MSE LWR; Qfilter	50 l.ex. MSE LWR; Qfilter	100 l.ex. MSE LWR; Qfilter	300 l.ex. MSE LWR; Qfilter
RoboY1	3.690 ; 3.421	1.201 ; 0.933	0.298 ; 0.196	0.019 ; 0.018

4.2 Population Dynamics Domain

The last domain models a dynamic behavior of an aquatic ecosystem that involves populations of zooplankton and phytoplankton, and inorganic nutrient nitrogen that are denoted by variables Zoo, $Phyto$ and Nut, respectively. The model assumes closed ecosystem with no inflow and consists of two consumption interactions. Namely, phytoplankton consumes nitrogen, and zooplankton consumes phytoplankton. This results in complex time behavior of the variables.

Our learning task is to predict the change in zooplankton $ZooChange(t)$, i.e. the difference between the zooplankton population at the next and the current time point ($ZooCh(t) = Zoo(t+1) - Zoo(t)$), given the values of zooplankton, phytoplankton, and nutrient at the current time point. We used experimental data that was kindly provided by Ljupčo Todorovski and Sašo Džeroski who previously experimented in this domain and give a more elaborate description of the domain [13]. The experimental data was generated by the following differential equations model:

$$\begin{aligned} \dot{Nut} &= 2 - Phyto\ Nut \\ \dot{Phyto} &= 0.1 - \frac{Phyto}{7} - \frac{Phyto}{5} + 0.7\,PhytoNut - 0.5\,\frac{Zoo\ Phyto}{Phyto + 0.5} \\ \dot{Zoo} &= -0.1\ Zoo + 0.25\,\frac{Zoo\ Phyto}{Phyto + 0.5} \end{aligned} \tag{3}$$

In contrast to other experimental domains we do not use a qualitative model that would completely correspond to the actual numerical behavior of the population dynamics model. Instead we use a heuristic qualitative tree given in Figure 3. This qualitative tree was obtained by qualitative abstraction of $\dot{Zoo}$ in Equation 3 and assumes constant values of the variables between the current and the next time point. It is just an approximate qualitative model an

expert might give and has the following interpretation. Since zooplankton feeds on phytoplankton, a larger phytoplankton population enables a bigger positive change in zooplankton. The change of zooplankton is also positively related to the zooplankton population, since the growth rate of a population is positively related to the size of the population. But if the phytoplankton population is too small (below 0.33 in qualitative tree in Figure 3) to provide enough food for zooplankton, then the change in zooplankton will be negatively related to the zooplankton population.

The data consists of ten traces generated by simulating a numerical model from ten randomly chosen triples of starting values for variables Zoo, $Phyto$ and Nut. Each simulation lasts for 100 time steps and gives 100 examples, each example being described with attributes $Nut(t)$, $Phyto(t)$ and $Zoo(t)$. The class variable $ZooCh$ was computed as the difference in zooplankton population between two consecutive points in time, i.e. $ZooCh(t) = Zoo(t+1) - Zoo(t)$. The learning examples were randomly selected from the first five traces, and the test examples were randomly selected from second five traces. We used 100 learning and 100 test examples. The results in Table 1 are averages of learning from ten random selections of examples. These results show that even an approximate qualitative model can help Qfilter to improve numerical accuracy.

In the experiments with Weka [4] implementation of M5 regression and model trees [2], qualitative errors were even more obvious, as illustrated also in [1]. For this reason the accuracy improvements of Qfilter with respect to model and regression trees were usually bigger.

5 Conclusions

We presented a novel approach to numerical machine learning called Qfilter. Qfilter is a numerical regression method that can take into account qualitative background knowledge expressed as a qualitative tree with qualitatively constrained functions in the leaves of the tree. As qualitative domain knowledge is often available in practice, Qfilter's ability to exploit such knowledge should be beneficial in many applications. One desirable consequence of using such qualitative knowledge is improved accuracy of numerical predictions. Another desirable property is that the resulting numerical regression model is qualitatively consistent with known qualitative relations in the domain of application.

There are several directions in which Qfilter can be extended. As noted in Section 3.3, a possible improvement is to use the confidence estimate in numerical prediction provided by the numerical predictor to change less the class values that have higher confidence estimate. Experiments with using the size of confidence intervals provided by LWR show that this can additionally improve Qfilter accuracy. Qfilter as presented in this paper, requires a numerical learner and it does not provide an explicit model. However, the quadratic programming approach can easily be extended to induce a piecewise linear model that is consistent with a given qualitative model. Another interesting point is that Qfilter finds minimal sum of squared changes of class values to achieve consistency with

a given qualitative model. In this respect it gives the error of numerical data w.r.t. qualitative model or vice versa and provides a bridge between qualitative and numerical models.

In the experiments in several domains, Qfilter always improved the accuracy of numerical predictions compared to standard regression methods. Improvements in accuracy were observed even in cases when the qualitative constraints applied were only approximate. In the experiments, the improvements were observed consistently when varying the amount of learning examples and the degree of noise in the data. In this paper we assumed that qualitative trees are given. An appealing alternative would be to use induced qualitative trees. QUIN, depending on the noise, often induced similar qualitative trees as used here.

Acknowledgements

The work reported in this paper was partially supported by the European Fifth Framework project Clockwork and the Slovenian Ministry of Education, Science and Sport.

References

1. Šuc, D., Vladušič, D., Bratko, I.: Qualitatively faithful quantitative prediction. In: Proceedings of the 18th International Joint Conference on Artificial Intelligence. (2003) August, 2003, Acapulco, Mexico.
2. Quinlan, J.: Learning with continuous classes. In: Proc. of the 5th Australian Joint Conference on Artificial Intelligence, Singapore, World Scientific (1992) 343–348
3. Atkeson, C., Moore, A., Schaal, S.: Locally weighted learning. Artificial Intelligence Review **11** (1997) 11–73
4. Witten, I., Frank, E.: Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations, Chapter 8. Morgan Kaufmann, San Francisco (2000) 265–320
5. Šuc, D.: Machine Reconstruction of Human Control Strategies. PhD thesis, Faculty of Computer and Information Sc., University of Ljubljana, Slovenia (2001)
6. Šuc, D., Bratko, I.: Induction of qualitative trees. In De Raedt, L., Flach, P., eds.: Proc. of the 12th European Conf. on Machine Learning, Springer (2001) 442–453
7. Šuc, D., Bratko, I.: Qualitative reverse engineering. In Sammut, C., Hoffmann, A., eds.: Proc. of the 19th International Conf. on Machine Learning, Morgan Kaufmann (2002) 610–617
8. Forbus, K.: Qualitative process theory. Artificial Intelligence **24** (1984) 85–168
9. Kuipers, B.: Qualitative simulation. Artificial Intelligence **29** (1986) 289–338
10. The MathWorks, I.: Matlab software. (2003) http://www.mathworks.com.
11. Coleman, T.F., Li, Y.: A reflective Newton method for minimizing a quadratic function subject to bounds on some of the variables. SIAM Journal on Optimization **6** (1996) 1040–1058
12. Gill, P.E., Murray, W., Wright, M.H.: Quadratic programming. In: Practical Optimization. Academic Press, London, England (1981) 177–184
13. Todorovski, L., Džeroski, S.: Using domain knowledge on population dynamics modeling for equation discovery. In: Proceedings of the 12th European Conference on Machine Learning, Springer (2001) 478–490